



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/900,060	07/06/2001	Travis J. Muhlestein	MSFT115921	7821

26389 7590 02/07/2006

CHRISTENSEN, O'CONNOR, JOHNSON, KINDNESS, PLLC
1420 FIFTH AVENUE
SUITE 2800
SEATTLE, WA 98101-2347

EXAMINER

VU, TUAN A

ART UNIT PAPER NUMBER

2193

DATE MAILED: 02/07/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.		Applicant(s)	
	09/900,060		MUHLESTEIN ET AL.	
	Examiner		Art Unit	
	Tuan A. Vu		2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 09 November 2005.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1 and 3-17 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1 and 3-17 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. This action is responsive to the Applicant's submission filed 11/9/2005.

As indicated in Applicant's submitted response, claims 1 and 9 have been amended.

Claims 1, 3-17 are pending in the office action.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

3. Claims 1, 3-17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Foody et al., USPN: 5,732,270 (hereinafter Foody) , in view of Katchabaw et al., "Making Distributed Applications Manageable Through Instrumentation", The Journal of Systems and Software 45, 1999 (hereinafter Katchabaw).

As per claim 1, Foody discloses a method for providing access to an external data access from within a managed code runtime computing system (Fig. 1), wherein the managed code environment provides a common language runtime engine (e.g. *interpreted, compiled technology, without disruption* --col. 7, line 39-58; Fig. 6-7; Fig 12C – Note: 'universal object' - enabling heterogeneous constructs language to cooperate - with creation and dynamic incorporation of application object classes or instances so they can be interpreted or executed in running applications **reads on** *common language runtime* where intermediate code can be interpreted, or native code executed) that compiles the intermediate language (e.g. col. 20, line

Art Unit: 2193

50 to col. 22, line 13) of an application to produce native machine instructions, the method comprising:

providing an external data access interface, - client application program interface - within said runtime environment (e.g. *Dynamic Invocation Interface* - col. 8, lines 14-15; *single API* - col. 11, lines 22-25; *Proxy object, the System* - Fig. 1; Fig. 10; *proxies ... in process .. the system* - col. 19, lines 23-39; *OSA, application* - col. 19, lines 51 to col. 20, line 3 - Note: OSA and remote calls reads on client application accessing Corba/COM service data);

receiving a request at said software components access interface for an external data access available outside of said runtime environment (e.g. step 42 - Fig 8; *first object, how to access ... call its methods, set its properties, handle exceptions* - col. 8, lines 48-61; Fig. 12c - Note: methods, properties, exceptions read on an external data access from outside the runtime environment);

transmitting a request for said external data access to an data source external to said runtime environment (e.g. *intercepted by the system, the system forwards* - col. 8, lines 58-65; col. 17, lines 5-35; *COM server 78* - Fig. 12C; *OSA*, col. 9, lines 5-47; *VfunctionData* - col. 11, lines 15-39; col. 15 lines 16-34 - Note: the system intercepting the proxy directions to access the real object via COM service is equivalent to transmitting a request for data to an external data access external source), using the API (e.g. *single API* - col. 11, lines 22-25)

receiving a response to said request (e.g. Fig. 9; *COM server 78* - Fig. 12C; *queries the objects, querying... information* -col. 10, lines 35-51; step 702 - Fig. 7C - Note: retrieving code or objects from additional service reads on receiving a response to request for object retrieval;

Art Unit: 2193

and Expose class Factory and return results for Factory class construction reads on receiving response to request);

converting said response to a format compatible with said runtime environment (e.g. col. 13, line 8 to col. 14, line 49; Fig. 5,6-7b; step 704 – Fig. 7C); and

responding to said request with converted response (e.g. step 51, 52 – Fig. 8; Fig. 7c).

But Foody does not expressly disclose that the external data being accessed are client instrumentation data. Foody, however, teaches providing of class objects and methods as tailored according to a proxy and wrapper paradigm using description in order to provide data making up the required software project or managed process of a developer runtime application with respect to remote database storage (e.g. Fig. 12-13; col. 16 line 54 to col. 17, line 4; col. 10, line 10 to col. 11, line 25), e.g. manipulating a code structure or class object in order to insert modifications/methods in order to effect data retrieval like remote retrieval of OSA information or to set up exception calls, thus teaches the type of data reminiscent of instrumentation activities data. Foody also teaches client interface code in conjunction distributed communication and adapter service like Corba/COM requiring remote calls to provide response for a requesting developers provided with conversion interface (see col. 3, line 8 to col. 22; Fig. 12c). In a system to retrieve application data from an external source to a client application runtime, e.g. requesting developers -- like that of Foody, Katchabaw discloses providing instrumentation components communicating (e.g. Fig. 1-2, pg. 83; ch. 3-5, 6.1-2, pg. 82-93) an agent and a instrumentation manager interface with managed process, the managed process having client interface code (ch. 4.2, 4.2.1; pg. 86-87 – Note: client dynamic code to communicate with agent further reads on client API) have been obvious for one of ordinary skill in the art at the time the

Art Unit: 2193

invention was made to modify to the multi-system code component assembling and customized delivery by Foody so to expand it into instrumentation components gathering and retrieval as taught by Katchabaw in order to support objects creation compliance, exceptions handling, and error checking as suggested by Foody (see Fig. 15) in order to provide a fault-free code for integration in the user's environment.

As per claim 3, Foody discloses converting object created in accordance with required configuration of runtime object (e.g. col. 15, line 8 to col. 16, line 28 – Note: factoring object according to specification of a dynamic proxy is equivalent to converting object model that is compatible with runtime environment of the requesting user; Fig. 6-7; col. 18, lines 7-36).

As per claim 4, Foody discloses returning a object (e.g. *Export Framework* – Fig. 2; col. 7, lines 39-59), a Factory (col. 16, lines 48-64; col. 17, line 55-66 – Note: creating a object by OSA from putting together classes and methods as in Fig. 5-7, 13-14, is equivalent to packaging properties of management object through instrumentation data source)

As per claims 5 and 6, Foody discloses exception object (e.g. *exceptions* - col. 10, lines 10-25; *VExceptionData* – Fig. 15; step 707 – Fig. 7C; *kVUsageProtected*, *kVUsageDoNotCache* – Fig. 15).

As per claims 7 and 8, Foody discloses a computer-readable medium (col. 235, lines 61-64)

As per claim 9, Foody discloses a method for accessing an external data access from within a managed code runtime computing environment wherein the managed code environment provides a common language runtime engine (e.g. *interpreted*, *compiled technology*, *without disruption* --col. 7, line 39-58; Fig. 6-7; Fig 12C – Note: 'universal object' - enabling

Art Unit: 2193

heterogeneous constructs language to cooperate - with creation and dynamic incorporation of application object classes or instances so they can be interpreted or executed in running applications **reads on common language runtime** where intermediate code can be interpreted, or native code executed) that compiles the intermediate language (e.g. col. 20, line 50 to col. 22, line 13) of an application to produce native machine instructions, comprising:

receiving a request to construct a management object comprising said system software components(step 42 – Fig 8);

in response to said request, querying an application data access interface within said application – or managed code - runtime environment for said software components (e.g. Fig. 2; com Dll ‘in Process’ – Fig. 12a), using a manage code application interface (e.g. *single API* – col. 11, lines 22-25);

determining whether said software components was successfully returned (Fig. 11-12 – Note: remote calls to COM server inherently discloses determining whether software components being retrieved are successfully returned; step 702-703 Fig 7C; *return errors* - col. 12, lines 39-47); and

in response to said successful return, constructing said management object and populating said object with requested software components (e.g. Fig. 7C, 9, 13-15 – Note: traversing class to create object class and methods for creating a object from the Factory is equivalent to constructing an management object and populating it with software objects).

But Foody does not expressly disclose that the application data being accessed/requested from the runtime environment are instrumentation data; but this limitation has been addressed in claim 1 above.

As per claim 10, Foody discloses binding a management class to an instance of management object (Fig. 8, 13-15)

As per claim 11, Foody discloses identifying a Namespace (e.g. col. 10, lines 10-39)

As per claim 12, Foody discloses management path object (e.g. *path ...information 68* – Fig. 10; *location ...framework* – Fig. 2)

As per claim 13, Foody discloses interface calls options to access class objects (e.g. *DSOM, SOM, COM* - col. 9, lines 51-61; *CORBA, DII* – col. 10, lines 32-48 – Note: providing different ways to access object is equivalent to providing access connecting options)

As per claim 14, Foody discloses exception when constructing a class; but does not provide throwing a management exception; but in view of the creating of class to access data using object-oriented programming constructs (see Appendix VViewNameSpace, Exception - col. 61-126), the suggestion as to throw exception upon non-success using class definition to access a management object, e.g. the namespace, is shown. Official notice is taken that using object-oriented like Java to implement access of data via a communication link and thus throw an exception call when a communication occurs was a well-known concept at the time the invention was made. Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide code as suggested by Foody so to implement communication with the management service by COM or OSA such that exception are thrown when such communicating incurs errors as suggested in Foody's Appendix and taught by well known practices.

As per claim 15, see Foody Appendix (e.g. *VcrCallException, Index* - col. 53-54)

As per claims 16 and 17, refer to the rationale of claims 7-8, respectively.

Response to Arguments

4. Applicant's arguments filed 11/9/2005 with respect to claims 1-17 have been considered but are not persuasive. Following are the Examiner's remarks in regard thereto.

(A) Applicants have submitted that Foody in the cited portions do not teach instrumentation client API within a managed code runtime environment; nor does Foody teach receiving a request for such instrumentation data outside said environment (Appl Rmrks, pg. 11, middle, pg. 12, top para). The *managed code runtime environment* of Claim 1 as recited amounts to an environment wherein a common language runtime engine compiles the intermediate language of an application to produce native machine instructions. The rejection has mapped this limitation with a universal object wherein object from application can be retrieved/located to be incorporated and interpreted or compiled for execution within running applications without disrupting the application (e.g. *interpreted, compiled technology, without disruption* --col. 7, line 39-58; Fig 12C). By 'universal object' as by Foody, it is to be perceived as an environment to enable cooperation of objects or classes that would otherwise need compilation in their own language environment, like DSOM or ORBIX, COM; but owing to the common language interoperability of Foody's wrapper/proxy system (see Foody's BACKGROUND: cols. 4-6), these heterogeneous language objects can be incorporated and work for the running applications dynamically, i.e. executed in native form or interpreted. The cited portions of Foody have hence disclosed the managed object runtime environment. Further, Foody teaches an application wherein specialized interface adapter classes and methods appearing as a single API are instantiated via a framework object in order to retrieve external components retrieved from COM server (see col. 11, lines 15-32; Fig. 12C); and in converting the retrieved data using a proxy

Art Unit: 2193

readaptation methods (col. 15 lines 16-56) in order to put forth VFunction data thus assembled into a execution stack (e.g. Fig. 7c; step 51, 52 – Fig. 8). And based on Foody's COM/server retrieved data being used to adapt to the application runtime code format and for detecting fault therein, Foody's data are suggestive of instrumentation used in debugging or code fault detection; and Katchabaw for having a COM/agent interface approach similar to Foody COM/proxy approach, is brought in to enable why the teaching of getting instrumentation data would have been obvious. Hence Foody in conjunction with Katchabaw has met all the limitations of claim 1; and since this is a combination of teaching, the arguments have to put forth why the rationale for combining as set forth in the rejection would be inappropriate. In other words, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986).

(B) The Applicants have submitted that managed code is executed by a 'common language runtime' engine rather than operating system; that Foody fails to teach 'compiles intermediate language into native instructions' (Appl. Rmrks, pg. 11, bottom). In reply, it is noted that the claim does not enforce any teaching that compels that the recited client application managed code runtime excludes operating system level service code; nor does the claim necessarily preclude Foody's adaptation/linking of reuse code in order to have them either linked into executable or interpreted for further readaptation (e.g. col. 20, line 50 to col. 22, line 13) from reading on the *intermediate code* limitation. That is, Foody's environment provides objects and classes to be incorporated in the environment in two forms, a form to be interpreted and a form to be executed (see Foody, *intermediate representation* - col. 20, line 50 to col. 22, line 13), and

Art Unit: 2193

this fulfills the above limitation. It is also noteworthy to put forth that while the arguments raised the compilation of intermediate code, the body of the claim as a whole does not provide any insight as to how this compilation is implemented, notwithstanding the fact that on the outset the claim recites a client system being a runtime environment, which makes it unclear as to how a compilation undertaking and a runtime can coexist without any clear description to that effect. Moreover, Foody teaches using DLL and relocation type of interface calls to retrieve intermediate objects from reuse repository and to set them up in a OSA-based proxy engine with a *VfunctionData* structure to enable their call making/readapting in order to have them type-checked, linked/loaded for execution in their native capability (see Fig. 6-7). This also reads on compiling intermediate form into executable native code. Foody teaches COM DLL (see col. 19); and it was a very well known concept that application must use API in order to invoke COM library such as Dlls, if not saying that COM DLL for requiring existence of API has made APIs an inherent teaching by Foody. The limitation phrased as ‘instrumentation client API’ is not described in more insightful terms so as to distinguish it from Foody’s API for adapting different parts to be located and via a proxy being incorporated for the runtime of the client application. The instrumentation aspect of objects being retrieved has been set forth in the rejection as to render the instrumentation characteristic of objects to retrieve by Foody’s proxy construction obvious when combining Foody with the DCE framework of Katchabaw. The interpretation of the limitation recited as “instrumentation client API” is that it amounts to the steps being recited, namely *receiving a request ... , transmitting a request..., receiving a response ... and converting said response*; and the rejection has mapped each of these steps appropriately.

Art Unit: 2193

(C) Applicants have submitted that Foody does not teach providing access to instrumentation data from within a managed code runtime environment and that Foody's proxy is a foreign object outside such environment (Appl. Rmrks, pg. 12, 2nd para). Foody's framework can be developed by one user using OSA objects and selectively creating proxy within application (see col. 11, lines 53-67); this is done within one framework using remote libraries or information via query by interfacing with Corba (col. 10, lines 10-49). The paradigm for data retrieval between a framework's user (with a proxy integral to the framework application needs) and a remote source of information storage via a query depicts a client/server communication, i.e. the proxy being at the developer or client machine, notably when there is no mention of a server proxy in Foody's framework. Hence, the argument is deemed not convincing. The instrumentation data limitation has been addressed above.

(D) Applicants have submitted that nowhere Katchabaw teaches a instrumentation client API within a managed code runtime environment (Appl. Rmrks, pg. 13). The instrumentation data limitation has been addressed above; the client API has been addressed in sections A and B above. And since Katchabaw has been brought in to address a retrieval of instrumentation data in a similar framework to Foody, the argument against any deficiency in Katchabaw without considering the teaching of Foody would have been inappropriate. As already proffered in the previous Office Action, it is noted that the claim does not recite specific teaching that would necessarily preclude any of the alleged findings concerning Katbachaw's Manager or Agent component as thus put forth. The rejection has pointed to parts of Katbachaw reference that also teach interface code for accessing another interface to a management component for getting instrumentation data (see ch. 4.2.1 pg. 86). Whether or not Katchabaw's Agent component or

Art Unit: 2193

the Manager component is included in the same machine as the managed process for which instrumentation data are to be retrieved via the Agent, the fact that the managed objects have interface classes and methods suffice to have met 'instrumentation client API' as interpreted from the claim. Until the claim provides further clarification on how this API consists of, the limitations in the claim are treated based on broad interpretation as set forth above (see section B); thus the arguments about a agent being not in a same environment seems unjustified, i.e. not persuasive or moot. Further, the applicants fail to acknowledge that the Katbachaw reference is only used to provide an instrumentation type of data in the RPC/Corba paradigm also as discussed by Foody. Hence Foody in conjunction with Katchabaw has met all the limitations of claim 1; and since this is a combination of teaching, the arguments have to put forth why the rationale for combining as set forth in the rejection would be inappropriate. In other words, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986). Besides, the claim as recited does not preclude an Agent component interface like Katbachaw's code operating just outside the managed application code; nor does any teaching in Katbachaw dictate that the Agent code is located in a machine other than that of the managed code, which appears not to be the case from Fig. 2.

(E) As for the arguments concerning claim 9, these are referred back to the above sections.

Thus, for the above reasons, the claims will stand rejected as set forth in the office action.

Conclusion

Art Unit: 2193

5. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).


A shortened statutory period for reply to this final action is set to expire **THREE MONTHS** from the mailing date of this action. In the event a first reply is filed within **TWO MONTHS** of the mailing date of this final action and the advisory action is not mailed until after the end of the **THREE-MONTH** shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than **SIX MONTHS** from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.


KAKALI CHAKI
SUPERVISOR MINER
TEC: 2100